CLAIMS

1. A method comprising:

forming a scale-independent logical model of an application to be implemented by a distributed computer system, the model having multiple components representing logical functions of the application; and

converting individual model components into one or more instances representative of physical resources that are used to implement the logical functions.

2. A method as recited in claim 1, wherein each instance represents a corresponding physical resource in a one-to-one relationship.

3. A method as recited in claim 1, wherein:

the distributed computer system comprises multiple computing nodes;

the model components comprise a module that is representative of a functional behavior of the application; and

an instance of the module corresponds to a node of the distributed computer system.

4. A method as recited in claim 1, further comprising allocating the physical resources to implement the instances.

5. A method as recited in claim 1, further comprising allocating the physical resources in real time to implement the instances.

6. A method as recited in claim 1, further comprising storing the instances in a database.

7. A method as recited in claim 6, further comprising querying the database to determine a current configuration of the application.

8. A computer-readable medium storing computer-executable instructions that, when executed on a computer, perform the method of claim 1.

9. A method comprising:

constructing an application for a distributed computer system according to a logical model, the logical model having multiple components representing logical functions of the application;

monitoring operation of the application during runtime; and

automatically deploying resources of the distributed computer system to the application as operation conditions change.

10. A method as recited in claim 9, wherein the application comprises an Internet service.

11. A method as recited in claim 9, wherein the constructing comprises creating one or more instances of each model component, the instances specifying physical resources used to implement the logical functions.

12. A method as recited in claim 11, further comprising storing the instances in a database.

13. A method as recited in claim 12, further comprising querying the database to determine a current configuration of the application.

14. A method as recited in claim 9, wherein the monitoring comprises receiving events regarding operating conditions of computer nodes in the distributed computer system and evaluating the events against a policy.

15. A method as recited in claim 9, wherein the distributed computer system comprises a plurality of interconnected computer nodes and the logical model comprises a module that is representative of a behavior of the application, and the deploying comprises:

creating one or more instances of each module component of the logical module, the instances specifying physical resources used to implement the behavior of the application; and

allocating a node of the distributed computer system for each instance of each module.

16. A method as recited in claim 9, further comprising tracking the resources that are deployed to the application and available resources that have not yet been deployed.

17. A computer-readable medium storing computer-executable instructions that, when executed on a computer, perform the method of claim 9.

18. A method comprising:

maintaining a logical model of an application for a distributed computer system, the logical model having multiple components representing logical functions of the application;

creating one or more instances of each component in the logical model; and

allocating resources of the distributed computer system to implement each of the instances.

19. A method as recited in claim 18, wherein the creating comprises producing multiple identical instances of a component in the logical model to accommodate operating conditions for the logical function represented by the component.

20. A method as recited in claim 18, wherein the distributed computer system comprises a plurality of interconnected computer nodes and the logical model comprises a module that is representative of a behavior of the application, the allocating comprising allocating a node for each instance of each module in the logical model.

21. A method as recited in claim 18, further comprising recording the instances in a database.

22. A method as recited in claim 21, further comprising querying the database to determine a current configuration of the application.

23. A method as recited in claim 18, further comprising tracking the resources that are allocated and correlating the resources with the instances for which the resources are allocated.

24. A computer-readable medium storing computer-executable instructions that, when executed on a computer, perform the method of claim 18.

25. A method comprising:

maintaining a logical model of an Internet service hosted on a plurality of interconnected computer nodes, the logical model having modules representing logical functions of the Internet service;

creating one or more instances of each module in the logical model;

allocating a computer node for each corresponding instance; and

configuring each computer node to perform the logical functions represented by the module from which the corresponding instance is created.

26. A method as recited in claim 25, wherein the configuring comprises loading software onto the computer node.

27. A method as recited in claim 25, wherein the configuring comprises downloading software from a remote location to the computer node via a network.

28. A method as recited in claim 25, wherein the configuring comprises:

initializing the computer node by installing a platform software; and

loading a software program that performs the logical functions associated with the instance.

29. A method as recited in claim 25, further comprising tracking the instances that are created in a database.

30. A method as recited in claim 25, further comprising:

adding a new instance of a particular module;

allocating a new computer node for the new instance; and

loading software onto the new computer node that performs the logical functions represented by the particular module.

31. A method as recited in claim 25, further comprising:

removing a particular instance of a particular module;

deallocating a computer node associated with the particular instance; and

returning the computer node to a pool of available computer nodes.

32. A computer-readable medium storing computer-executable instructions that, when executed on a computer, perform the method of claim 25.

33. A system to deploy an application for a distributed computer system having a plurality of computer nodes, the system comprising:

a logical model of the application, the logical model having multiple components representing logical functions of the application; and

a core converter to create one or more instances of the model components and allocate computer nodes of the distributed computer system for the instances to implement the logical functions represented by the model components from which the instances are created.

34. A system as recited in claim 33, wherein the core converter comprises:

a service running state to track the instances created for the model components; and

a resource manager to track the computer nodes available to be allocated.

35. A system as recited in claim 33, wherein the core converter comprises a loader to load software on the computers nodes to implement the logical functions.

36. A system as recited in claim 33, wherein the core runtime converter comprises:

a service running state to track the instances created for the model components;

a resource manager to track the computer nodes available to be allocated; and

a loader to load software onto the new computer node, the software being executable on the computer node to implement the logical functions represented by the particular model component.

37. A system as recited in claim 33, further comprising a management policy to monitor operation of the application and to specify when new instances of the model components are to be created or removed based on the operation of the application.

38. A system as recited in claim 37, wherein the management policy listens to events generated by the computer nodes as a way to monitor operation of the application.

39. A model conversion system comprising:

a service running state to maintain a logical model of a service application to be implemented by software distributed across a plurality of computer nodes, the logical model having multiple components representing logical functions of the application;

a resource manager to allocate computer nodes for the instances; and

a loader to load various software onto the computer nodes allocated by the resource manager, the software being executable on the computer nodes to implement the logical functions represented by the model components from which the instances are derived.

40. A system as recited in claim 39, wherein the service running state tracks the instances.

41. A system as recited in claim 39, wherein the resource manager tracks whether the computer nodes are allocated or unallocated.

42. A system as recited in claim 39, further comprising a management policy to monitor operation of the application and to specify when new instances of the model components are to be created or removed based on the operation of the application.

43. A system as recited in claim 42, wherein the management policy listens to events generated by the computer nodes as a way to monitor operation of the application.

44. A system as recited in claim 39, further comprising a node loader resident at each of the computer nodes to install the software onto the computer nodes.

45. A system comprising:

means for maintaining a scale-independent logical model of a service application to be implemented by software distributed across a plurality of computer nodes, the logical model having multiple components representing logical functions of the application;

means for creating one or more instances of the model components according to a desired scale of the service application; and

means for allocating the computer nodes to associated instances of the model components, the computer nodes being configured to perform the logical functions represented by the components from which the instances are created.

46. A system as recited in claim 45, further comprising means for tracking the instances.

47. A system as recited in claim 45, further comprising means for tracking the allocated computer nodes.

48. A system as recited in claim 45, further comprising means for facilitating policy that specifies when and what instance of a particular model component is created.

49. A system as recited in claim 45, further comprising means for loading software onto the computer nodes following allocation, the software being executed on the computer nodes to perform the logical functions represented by the components from which the instances are created.

50. One or more computer-readable media comprising computer-executable instructions that, when executed on one or more processors, direct one or more computing devices to:

maintain a logical model of an application to be implemented by software distributed across a plurality of computer nodes, the logical model having multiple components representing logical functions of the application; and

convert the model components into one or more instances representative of physical resources used to implement the logical functions.

51. One or more computer-readable media as recited in claim 50, further comprising computer-executable instructions that, when executed on one or more processors, direct one or more computing devices to allocate the physical resources for each of the instances.

52. One or more computer-readable media as recited in claim 50, further comprising computer-executable instructions that, when executed on one or more processors, direct one or more computing devices to track the instances in a database.

53. A computer-readable storage medium storing a data structure, the data structure comprising:

a logical model of an application for a distributed computer system, the logical model having at least one module that represents a functional behavior of the application, at least one port that represents a communication access point for

the module, and at least one wire that represents a logical connection between the port of the module and a port of another module;

a first structure to store module information pertaining to one or more module instances of the module that correspond to physical resources used to implement the functional behavior represented by the module;

a second structure to store port information pertaining to one or more port instances of the port; and

a third structure to store wire information pertaining to one or more wire instances of the wire.

54. A computer-readable storage medium as recited in claim 53, wherein the module information in the first structure is correlated with the port information in the second structure to associate certain ports with certain modules.

55. A computer-readable storage medium as recited in claim 53, wherein the port information in the second structure is correlated with the wire information in the third structure to associate certain wires with certain ports.

56. A computer-readable storage medium as recited in claim 53, wherein the module information includes data fields selected from a group of fields including an identity of the module instance, an identity of the module in the logical model from which the module instance is created, an identity of a physical computer node on which the module instance is instantiated, an identity of a port for the module, and a protocol supported by the module.

57. A computer-readable storage medium as recited in claim 53, wherein the port information includes data fields selected from a group of fields including an identity of the port instance, an identity of the port in the logical model from which the port instance is created, a network address of a physical computer node on which the port instance is instantiated, an identity of a module with which the port is the communication access point, and a protocol supported by the port.

58. A computer-readable storage medium as recited in claim 53, wherein the wire information includes data fields selected from a group of fields including an identity of the wire instance, an identity of the wire in the logical model from which the wire instance is created, an identity of a port with which the wire is coupled, and a protocol supported by the wire.